arXiv:1304.1192v1 [cs.LG] 3 Apr 2013

Efficient Distance Metric Learning by Adaptive Sampling and Mini-Batch Stochastic Gradient Descent (SGD)

Qi Qian[†], Rong Jin[†], Jinfeng Yi[†], Lijun Zhang[†] and Shenghuo Zhu[‡]
[†]Department of Computer Science and Engineering
Michigan State University, East Lansing, MI, 48824, USA
[‡]NEC Laboratories America, Cupertino, CA, 95014, USA
{qianqi, rongjin, yijinfen, zhanglij}@cse.msu.edu, zsh@nec-labs.com

ABSTRACT

Distance metric learning (DML) is an important task that has found applications in many domains. The high computational cost of DML arises from the large number of variables to be determined and the constraint that a distance metric has to be a positive semi-definite (PSD) matrix. Although stochastic gradient descent (SGD) has been successfully applied to improve the efficiency of DML, it can still be computationally expensive because in order to ensure that the solution is a PSD matrix, it has to, at *every iteration*, project the updated distance metric onto the PSD cone, an expensive operation. We address this challenge by developing two strategies within SGD, i.e. mini-batch and adaptive sampling, to effectively reduce the number of updates (i.e., projections onto the PSD cone) in SGD. We also develop hybrid approaches that combine the strength of adaptive sampling with that of mini-batch online learning techniques to further improve the computational efficiency of SGD for DML. We prove the theoretical guarantees for both adaptive sampling and mini-batch based approaches for DML. We also conduct an extensive empirical study to verify the effectiveness of the proposed algorithms for DML.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Distance Metric Learning, Stochastic Gradient Descent, Mini-Batch, Adaptive Sampling

1. INTRODUCTION

Distance metric learning (DML) is an important subject, and has found applications in many domains, including information retrieval [14], supervised classification [19], clustering [20], and semi-supervised clustering [6]. The objective of DML is to learn a distance metric consistent with a given set of constraints, namely minimizing the distances between pairs of data points from the same class and maximizing the distances between pairs of data points from different classes. The constraints are often specified in the form of must-links, where data points belong to the same class, and cannot-links, where data points belong to different classes. The constraints can also be specified in the form of triplets $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k)$ [19], in which $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to a class different from that of $\mathbf{x}_k$ and therefore $\mathbf{x}_i$ and $\mathbf{x}_j$ should be separated by a distance smaller than that between $\mathbf{x}_i$ and $\mathbf{x}_k$. In this work, we focus on DML using triplet constraints due to its encouraging performance [7, 18, 19].

The main computational challenge in DML arises from the restriction that the learned distance metric must be a positive semi-definite (PSD) matrix, which is often referred as the *PSD constraint*. Early approach [20] addressed the PSD constraint by exploring the technique of semi-definite programming (SDP) [2], which unfortunately does not scale to large and high dimensional datasets. More recent approaches [7, 18] addressed this challenge by exploiting the techniques of online learning and stochastic optimization, particularly stochastic gradient descent (SGD), that only needs to deal with one constraint at each iteration. Although these approaches are significantly more efficient than the early approach, they share one common drawback: in order to ensure that the learned distance metric is PSD, these approaches require, *at each iteration*, projecting the updated distance metric onto the PSD cone. The projection step requires performing the eigen-decomposition for a given matrix, and therefore is computationally expensive [1]. As a result, the key challenge in developing efficient SGD algorithms for DML is how to reduce the number of projections without affecting the performance of DML.

A common approach for reducing the number of updates and projections in DML is to use the non-smooth loss function. A popular choice of the non-smooth loss function is the hinge loss, whose derivative becomes zero when the input value exceeds a certain threshold. Many online learning

[1]The computational cost is $O(d^2)$ if we only need to compute the top eigenvectors of the distance metric and becomes $O(d^3)$ if all the eigenvalues and eigenvectors have to be computed for the projection step, where d is the dimensionality of the data.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$15.00.

algorithms for DML [7, 9, 16] take advantage of the non-smooth loss function to reduce the number of updates and projections. In [18], the authors proposed a structure preserving metric learning algorithm (SPML) that combines a mini-batch strategy with the hinge loss to further reduce the number of updates for DML. It groups multiple constraints into a mini-batch and performs only one update of the distance metric for each mini-batch. But, according to our empirical study, although SPML reduces the running time of the standard SGD algorithm, it results in a significantly worse performance for several datasets, due to the deployment of the mini-batch strategy.

In this work, we first develop a new mini-batch based SGD algorithm for DML, termed **Mini-SGD**. Unlike SPML that relies on the hinge loss, the proposed Mini-SGD algorithm uses a *smooth* loss function for DML. We show theoretically that by using a smooth loss function, Mini-SGD is able to achieve similar convergence rate as the standard SGD algorithm but with significantly less number of updates. The second contribution of this work is to develop a new strategy, termed **adaptive sampling**, for reducing the number of projections in DML. The key idea of adaptive sampling is to first measure the "difficulty" in classifying a constraint using the learned distance metric, and then perform stochastic updating based on the classification difficulty. More specifically, given the distance metric M_t and triplet $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$, we first measure the difficulty in classifying the triplet $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$ by $\gamma_t = \ell'(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t)$, where $\ell(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t)$ is the loss function that measures the classification error. We then sample a binary variable Z_t with $\Pr(Z_t = 1) \propto \gamma_t$, and only update the distance metric when $Z_t = 1$. We refer to the proposed approach for DML as **AS-SGD** for short. Finally, we develop two **hybrid approaches**, termed **HA-SGD** and **HR-SGD**, that combine adaptive sampling with mini-batch to further improve the computational efficiency of SGD for DML. We conduct an extensive empirical study to verify the effectiveness and efficiency of the proposed algorithms for DML.

The rest of the paper is organized as follows: Section 2 reviews the related work on distance metric learning and stochastic gradient descent with reduced number of projection steps. Section 3 describes the proposed SGD algorithms for DML based on mini-batch and adaptive sampling. Two hybrid approaches are presented that combine mini-batch and adaptive sampling for DML. The theoretical guarantees for both mini-batch based and adaptive sampling based SGD are also presented in Section 3. Section 4 summarizes the results of the empirical study, and Section 5 concludes this work with future directions.

2. RELATED WORK

Many algorithms have been developed to learn a linear distance metric from pairwise constraints, where must-links include pairs of data points from the same class and cannot-links include pairs of data points from different classes ([21] and references therein). Besides pairwise constraints, an alternative strategy is to learn a distance metric from a set of triplet constraints $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t), t = 1, \ldots, N$, where $\mathbf{x}_i^t$ is expected to be closer to $\mathbf{x}_j^t$ than to $\mathbf{x}_k^t$. Previous studies [7, 18, 19] showed that triplet constraints could be more effective for DML than pairwise constraints.

Several online algorithms have been developed to reduce the computational cost of DML [7, 9, 12, 16]. Most of these methods are based on stochastic gradient descent. At each iteration, they randomly sample *one* constraint, and update the distance metric based on the sampled constraint. The updated distance metric is further projected onto the PSD cone to ensure that it is PSD. Although these approaches are significantly more scalable than the batch learning algorithms for DML [19], they suffer from the high computational cost in the projection step that has to be performed at *every* iteration. A common approach for reducing the number of projections is to use a non-smooth loss function, such as the hinge loss. In addition, in [18], the authors proposed a structure preserving metric learning (SPML) that combines mini-batch with the hinge loss to further reduce the number of projections. The main problem with the approach proposed in [18] is that according to the theory of mini-batch, it only works well with a smooth loss. Since the hinge loss is a non-smooth loss function, combining mini-batch with the hinge loss may result in a suboptimal performance. This is verified by our empirical study in which we observed that the distance metric learned by SPML performs significantly worse than that learned by the standard stochastic gradient descent method. We resolve this problem by presenting a new SGD algorithm for DML that combines mini-batch with a smooth loss, instead of the hinge loss.

Finally, it is worthwhile mentioning several recent studies proposed to avoid projections in SGD. In [13], the authors developed a projection free SGD algorithm that replaces the projection step with a constrained linear programming problem. In [17], the authors proposed a SGD algorithm with only one projection that is performed at the end of the iterations. Unfortunately, the improvement of the two algorithms in computational efficiency is limited, because they require computing, *at each iteration*, the minimum eigenvalue and eigenvector of the updated distance metric, an operation with $O(d^2)$ cost, where d is the dimensionality of the data.

3. IMPROVED SGD FOR DML BY MINI-BATCH AND ADAPTIVE SAMPLING

We first review the basic framework of DML with triplet constraints. We then present two strategies to improve the computational efficiency of SGD for DML, one by mini-batch and one by adaptive sampling. We present the theoretical guarantees for both strategies, and defer more detailed analysis to the appendix. At the end of this section, we present two hybrid approaches that combine mini-batch with adaptive sampling for more efficient DML.

3.1 DML with Triplet Constraints

Let $\mathcal{X} \subset \mathbb{R}^d$ be the domain for input patterns, where d is the dimensionality. For the convenience of analysis, we assume all the input patterns with bounded norm, i.e. $\forall \mathbf{x} \in \mathcal{X}, |\mathbf{x}|_2 \leq r$. Given a distance metric $M \in \mathbb{R}^{d\times d}$, the distance square between $\mathbf{x}_a$ and $\mathbf{x}_b$, denoted by $|\mathbf{x}_a - \mathbf{x}_b|_M^2$, is measured by

$$|\mathbf{x}_a - \mathbf{x}_b|_M^2 = (\mathbf{x}_a - \mathbf{x}_b)^\top M(\mathbf{x}_a - \mathbf{x}_b)$$

Let $\Omega = \{M : M \succeq 0, \|M\|_F \leq R\}$ be the domain for distance metric M, where R specifies the domain size. Let $\mathcal{D} = \{(\mathbf{x}_i^1, \mathbf{x}_j^1, \mathbf{x}_k^1), \ldots, (\mathbf{x}_i^N, \mathbf{x}_j^N, \mathbf{x}_k^N)\}$ be the set of triplet constraints used for DML, where $\mathbf{x}_i^t$ is expected to be closer to $\mathbf{x}_j^t$ than to $\mathbf{x}_k^t$. Let $\ell(z)$ be the convex loss function. Define $\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M)$ as

$$\begin{aligned}\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M) &= |\mathbf{x}_i^t - \mathbf{x}_k^t|_M^2 - |\mathbf{x}_i^t - \mathbf{x}_j^t|_M^2 \\ &= \left\langle M, (\mathbf{x}_i^t - \mathbf{x}_k^t)(\mathbf{x}_i^t - \mathbf{x}_k^t)^\top - (\mathbf{x}_i^t - \mathbf{x}_j^t)(\mathbf{x}_i^t - \mathbf{x}_j^t)^\top \right\rangle \\ &= \langle M, A_t \rangle\end{aligned}$$

where

$$A_t = (\mathbf{x}_i^t - \mathbf{x}_k^t)(\mathbf{x}_i^t - \mathbf{x}_k^t)^\top - (\mathbf{x}_i^t - \mathbf{x}_j^t)(\mathbf{x}_i^t - \mathbf{x}_j^t)^\top$$

Given the triplet constraints in $\mathcal{D}$ and the domain in Ω, we learn an optimal distance metric $M \in \mathbb{R}^{d \times d}$ by solving the following optimization problem

$$\min_{M \in \Omega} \quad \mathcal{L}(M) = \frac{1}{N} \sum_{t=1}^{N} \ell\left(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M)\right) \qquad (1)$$

The key idea of online DML is to update the distance metric based on one sampled constraint at each iteration. More specifically, at iteration t, it samples a triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$, and updates the distance metric M_t to M_{t+1} by

$$M_{t+1} = \Pi_\Omega\left(M_t - \eta \ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t)) A_t\right)$$

where $\eta > 0$ is the step size, $\ell'(\cdot)$ is the derivative and $\Pi_\Omega(M)$ projects a matrix M onto the domain Ω. The following proposition shows $\Pi_\Omega(M)$ can be computed in two steps, i.e. first projecting M onto the PSD cone, and then scaling the projected M to fit in with the constraint $\|M\|_F \leq R$.

Proposition 1. *[2] We have*

$$\Pi_\Omega(M) = \frac{1}{\max(\|M'\|_F / R, 1)} M'$$

where $M' = P(M)$ and $P(M)$ projects matrix M onto the PSD cone.

As indicated by Proposition 1, $\Pi_\Omega(M)$ requires projecting distance metric M onto the PSD cone, an expensive operation that requires eigen-decomposition of M.

Finally, to bound both the regret and the number of updates, in this study, we approximate the hinge loss by a smooth loss function

$$\ell(z) = \frac{1}{L} \log(1 + \exp(-L(z-1))) \qquad (2)$$

where $L > 0$ is a parameter that controls the approximation error: the larger the L, the closer $\ell(z)$ is to the hinge loss. Note that the smooth approximation of the hinge loss was first suggested in [23] for classification and was later verified by an empirical study in [22]. The key properties of the loss function $\ell(z)$ in (2) are given in the following proposition.

Proposition 2. *For the loss function defined in (2), we have*

$$\forall z \in \mathbb{R}, \quad |\ell'(z)| \leq 1, \; |\ell'(z)| \leq L\ell(z)$$

Compared to the hinge loss function, the main advantage of the loss function in (2) is that it is a smooth loss function. As will be revealed by our analysis, it is the smoothness of the loss function that allows us to effectively explore both the mini-batch and adaptive sampling strategies for more efficient DML without having to sacrifice the prediction performance.

Algorithm 1 Mini-batch Stochastic Gradient Descent (Mini-SGD) for DML

```
1: Input: triplet constraints $\{(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)\}_{t=1}^N$, step size $\eta$,
   mini-batch size $b$, and domain size $R$
2: Initialize $M_1 = I$ and $T = N/b$
3: for $t = 1, \ldots, T$ do
4:     Sample $b$ triplet constraints $\{(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s})\}_{s=1}^b$
5:     Update the distance metric by
           $M_{t+1} = \Pi_\Omega(M_t - \eta \nabla \ell_t(M_t))$
6: end for
7: return $\bar{M} = \frac{1}{T}\sum_{t=1}^T M_t$
```

3.2 Mini-batch SGD for DML (Mini-SGD)

Mini-batch SGD improves the computational efficiency of online DML by grouping multiple constraints into a mini-batch and only updating the distance metric once for each mini-batch. For brevity, we will refer to this algorithm as **Mini-SGD** in the rest of the paper.

Let b be the batch size. At iteration t, it samples b triplet constraints, denoted by

$$(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}), s = 1, \ldots, b,$$

and defines the mini-batch loss at iteration t as

$$\ell_t(M_t) = \frac{1}{b} \sum_{s=1}^{b} \ell\left(\Delta(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}; M_t)\right)$$

Mini-batch DML updates the distance metric M_t to M_{t+1} using the gradient of the mini-bach loss function $\ell_t(M)$, i.e.,

$$M_{t+1} = \Pi_\Omega(M_t - \eta \nabla \ell_t(M_t))$$

Algorithm 1 gives the detailed steps of Mini-SGD for DML, where step 5 uses Proposition 1 for computing the projection $\Pi_\Omega(\cdot)$.

The theorem below provides the theoretical guarantee for the Mini-SGD algorithm for DML using the smooth loss function defined in (2).

Theorem 1. *Let $\bar{M}$ be the solution output by Algorithm 1 that uses the loss function defined in (2). Let M_* be the optimal solution to (1). Assume $\|A_t\|_F \leq A$ for any triplet constraint. For a fixed $\delta \in (0,1)$, we have, with a probability $1 - 2\delta$:*

$$\begin{aligned}\mathcal{L}(\bar{M}) \leq \frac{\mathcal{L}(M_*)}{1 - 3\eta L A^2} &+ \frac{bR^2}{2(1 - 3\eta L A^2)\eta N} \\ &+ \frac{C_1 A^2 \eta}{(1 - 3\eta L A^2) N} \left[\log \frac{2N}{\delta b}\right]^2 \log \frac{m}{\delta}\end{aligned} \qquad (3)$$

where $m = \lceil \log_2 N \rceil$, and C_1 is an universal constant that is at most 32.

Figure 1 shows the reduction in the training error over the number of triplet constraints by the Mini-SGD algorithm on three datasets [2]. Compared to the standard SGD algorithm, we observe that Mini-SGD converges to a similar value of training error, thus validating our theorem empirically.

Remark 1 We observe that the second term in the upper bound in (3), i.e., $bR^2/[2(1-3\eta LA^2)\eta N]$, has a linear dependence on mini-batch size b, implying that the larger the b,

[2]The information of these datasets can be found in the experimental section.

Algorithm 2 Adaptive Sampling Stochastic Gradient Descent (AS-SGD) for DML

```
1: Input: triplet constraints $\{(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)\}_{t=1}^N$, step size $\eta$,
   and domain size $R$
2: Initialize $M_1 = I$
3: for $t = 1, \ldots, N$ do
4:    Sample a binary random variable $Z_t$ with
          $\Pr(Z_t = 1) = |\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t)|$
5:    if $Z_t = 1$ then
6:       Update the distance metric by
             $\tau_t = \mathrm{sign}(\ell'(\Delta(\mathrm{x}_i^t, \mathrm{x}_j^t, \mathrm{x}_k^t; \mathrm{M}_t))$
             $M_{t+1} = \Pi_\Omega (M_t - \eta\tau_t A_t)$
7:    end if
8: end for
9: return $\bar{M} = \frac{1}{N}\sum_{t=1}^N M_t$
```

the less accurate the distance metric learned by Algorithm 1. Hence, by adjusting parameter b, the size of mini-batch, we are able to make appropriate tradeoff between the prediction accuracy and the computational efficiency: the smaller the b, the more accurate the distance metric but with more updates and consequentially higher computational cost. Finally, it is worthwhile comparing Theorem 1 to the theoretical result for a general mini-batch SGD algorithm given in [8], i.e.

$$\mathcal{L}(\bar{M}) \leq \mathcal{L}(M_*) + O\left(\frac{1}{\sqrt{N}} + \frac{b^2}{N^2}\right) \tag{4}$$

It is clear that Theorem 1 gives a significantly better result when the optimal loss $\mathcal{L}(M_*)$ is small (i.e. when the triplet constraints can be well classified by the optimal distance metric M_*). In particular, when $\mathcal{L}(M_*) = O(b/N)$, the convergence rate given in Theorem 1 is on the order of $O(b/N)$ while the convergence rate in (4) is only $O(1/\sqrt{N})$.

3.3 Adaptive Sampling based SGD for DML (AS-SGD)

We now develop a new approach for reducing the number of updates in SGD in order to improve the computational efficiency of DML. Instead of updating the distance metric at each iteration, the proposed strategy introduces a random binary variable to decide if the distance metric M_t will be updated given a triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$. More specifically, it computes the derivative $\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))$, and samples a random variable Z_t with probability

$$\Pr(Z_t = 1) = |\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))|$$

The distance metric will be updated only when $Z_t = 1$. According to Proposition 2, we have $|\ell'(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))| \leq L\ell(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))$ for the smooth loss function given in (2), implying that a triplet constraint has a high chance to be used for updating the distance metric if it has a large loss. Therefore, the essential idea of the proposed adaptive sampling strategy is to give a large chance to update the distance metric when the triplet is difficult to be classified and a low chance when the triplet can be classified correctly with large margin. We note that an alternative strategy is to sample a triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$ base on its loss $\ell(\Delta(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t; M_t))$. We did not choose the loss as the basis for updating because it is the derivative, not the loss, that will be used by SGD for updating the distance metric. The detailed steps of adaptive sampling based SGD for DML is given in Algorithm 2. We refer to this algorithm as **AS-SGD** for short in the rest of this paper.

The theorem below provides the performance guarantee for AS-SGD. It also bounds the number of updates $\sum_{t=1}^T Z_t$ for AS-SGD.

THEOREM 2. *Let $\bar{M}$ be the solution output by Algorithm 2 that uses the loss function defined in (2). Let M_* be the optimal solution to (1). Assume $\|A_t\|_F \leq A$ for any triplet constraint. For a fixed $\delta \in (0,1)$, we have, with a probability $1 - 2\delta$:*

$$\mathcal{L}(\bar{M}) \leq \frac{\mathcal{L}(M_*)}{1 - 3\eta L A^2} + \frac{C_2}{(1 - 3\eta L A^2)N}\left(\frac{R^2}{\eta} + \eta + 1\right) \tag{5}$$

and

$$\sum_{t=1}^N Z_t \leq \frac{3}{2} L \sum_{t=1}^N \ell(M_t) + \frac{5}{2}\ln\frac{m}{\delta} \tag{6}$$

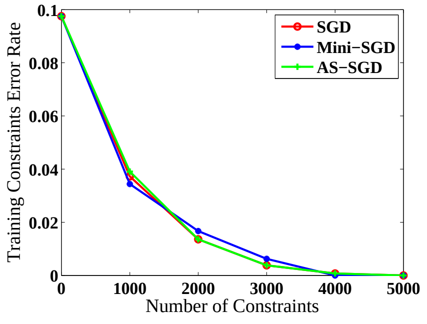
where

$$\begin{aligned} C_2 &= \max\left\{\frac{1}{2} + 16\ln\frac{m}{\delta}, \frac{5}{4}A^2\ln\frac{m}{\delta}, RA\ln\frac{2m}{\delta}\right\} \\ m &= \lceil \log_2(N^2) \rceil \end{aligned}$$

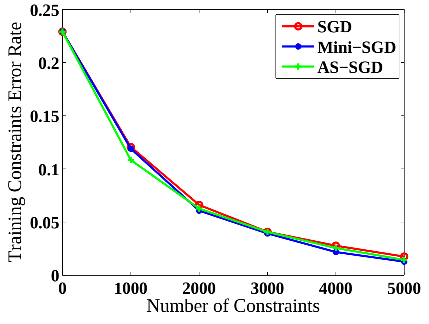
Remark 2 The bound given in (5) shares similar structure as that given in (3) except that it does not have mini-batch size b that can be used to make tradeoff between the number of updates and the classification accuracy. The number of updates performed by Algorithm 2 is bounded by (6). The dominate term in (6) is $O(\sum_{t=1}^N \ell(M_t))$, implying that Algorithm 2 will have a small number of updates if the learned distance metric M_t can classify the triplet constraint correctly at most iterations. In other words, the smaller the number of classification mistakes made by the learned distance metric M_t, the less number of updates will be performed by Algorithm 2. We validate the theorem by running the AS-SGD algorithm on three datasets. Figure 1 shows the reduction in the training error over the number of triplet constraints by AS-SGD and the standard SGD algorithm. We observe that AS-SGD converges to a similar value of training error as the full SGD algorithm.

3.4 Hybrid Approaches: Combine Mini-batch with Adaptive Sampling for DML

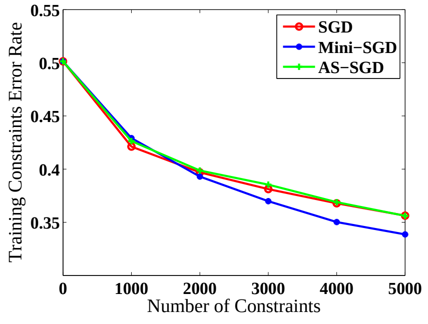
Since mini-batch and adaptive sampling improve the computational efficiency of SGD from different aspects, it is natural to combine them together for more efficient DML. Similar to the Mini-SGD algorithm, the hybrid approaches will group multiple triplet constraints into a mini-batch. But, unlike Mini-SGD that updates the distance metric for every mini-batch of constraints, the hybrid approaches follow the idea of adaptive sampling, and introduce a binary random variable to decide if the distance metric will be updated for every mini-batch of constraints. By combining the strength of mini-batch and adaptive sampling for SGD, the hybrid approaches are able to make further improvement in the computational efficiency of DML. Algorithm 3 highlights the key steps of the hybrid approaches.



(a) *semeion*



(b) *dna*



(c) *protein*

Figure 1: The convergence of different SGD algorithms

Algorithm 3 A Framework of Hybrid Stochastic Gradient Descent (Hybrid-SGD) for DML

```
1: Input: triplet constraints $\{(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)\}_{t=1}^N$, step size $\eta$,
   mini-batch size $b$, and domain size $R$
2: Initialize $M_1 = I$ and $T = N/b$
3: for $t = 1, \ldots, T$ do
4:     Sample $b$ triplets $\{\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}\}_{s=1}^b$.
5:     $\boxed{\text{Compute sampling probability } \gamma_t.}$
6:     Sample a binary random variable $Z_t$ with
           $\Pr(Z_t = 1) = \gamma_t$
7:     if $Z_t = 1$ then
8:         Update the distance metric by
               $\tau_t = 1/\gamma_t$
               $M_{t+1} = \Pi_\Omega(M_t - \eta\tau_t\nabla\ell_t(M_t))$
9:     end if
10: end for
11: return $\bar{M} = \frac{1}{T}\sum_{t=1}^T M_t$
```

One of the key steps in the hybrid approaches (step 5 in Algorithm 3) is to choose appropriate sampling probability γ_t for every mini-batch constraints $(\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}), s = 1, \ldots, b$. In this work, we study two different choices for sampling probability γ_t:

- The first approach chooses γ_t based on a triplet constraint randomly sampled from a mini-batch. More specifically, given a mini-batch of triplet constraints $\{\mathbf{x}_i^{t,s}, \mathbf{x}_j^{t,s}, \mathbf{x}_k^{t,s}\}_{s=1}^b$, it randomly samples an index s' in the range $[1, b]$. It then sets the sampling probability γ_t to be the derivative for the randomly sampled triplet, i.e.,

$$\gamma_t = |\ell'(\Delta(\mathbf{x}_i^{t,s'}, \mathbf{x}_j^{t,s'}, \mathbf{x}_k^{t,s'}; M_t))|$$

  We refer to this approach as **HR-SGD**.

- The second approach is based on the average case analysis. It sets the sampling probability as the average derivative measured by the norm of the gradient $\nabla\ell_t(M_t)$, i.e.,

$$\gamma_t = \frac{1}{W}\|\nabla\ell_t(M_t)\|_F$$

  where $W = \max_t \|\nabla\ell_t(M_t)\|_F$ and is estimated by sampling. We refer to this approach as **HA-SGD**.

Table 1: Statistics for the ten datasets used in our empirical study.

	# class	# feature	# train	# test
semeion	10	256	1,115	478
dna	3	180	2,000	1,186
isolet	26	617	6,238	1,559
tdt30	30	200	6,575	2,819
letter	26	16	15,000	5,000
protein	3	357	17,766	6,621
connect4	3	42	47,289	20,268
sensit	3	100	78,823	19,705
rcv20	20	200	477,141	14,185
poker	10	10	1,000,000	25,010

4. EXPERIMENTS

Ten datasets are used to validate the effectiveness of the proposed algorithms. Table 1 summarizes the information of these datasets. Datasets *dna*, *letter* [15], *protein* and *sensit* [10] are downloaded from LIBSVM [5]. Datasets *tdt30* and *rcv20* are document corpora: *tdt30* is the subset of tdt2 data [3] comprised of the documents from the 30 most popular categories and *rcv20* is the subset of a large rcv1 dataset [1] consisted of documents from the 20 most popular categories. We reduce the dimensionality of these document datasets to 200 by principle components analysis (PCA). All the other datasets are downloaded directly from the UCI repository [11]. For most datasets used in this study, we use the standard training/testing split provided by the original dataset, except for datasets *semeion*, *connect4* and *tdt30*. For these three datasets, we randomly select 70% of data for training and use the remaining 30% for testing; experiments related to these three datasets are repeated ten times, and the prediction result averaged over ten trials is reported. All experiments are implemented on a laptop with 8GB memory and two 2.50GHz Intel Core i5-2520M CPUs.

4.1 Parameter Setting

The parameter L in the loss function (2) is set to be 3 according to the suggestion in [23]. We set $N = 100,000$ for the number of iterations (i.e., the number of triplet constraints). To construct a triplet constraint at each iteration

t, we first randomly sample an example $(\mathbf{x}_i^t, y_i^t)$ from the training data; we then find two of its nearest neighbors $\mathbf{x}_j^t$ and $\mathbf{x}_k^t$, measured by Euclidean distance, from the training examples, with $\mathbf{x}_j^t$ sharing the same class label as $\mathbf{x}_i^t$ and $\mathbf{x}_k^t$ belonging to a class different from y_i^t. For Mini-SGD and the hybrid approaches, we set $b = 10$ for the size of mini-batch as in [18], leading to a total of $T = 10,000$ iterations for these approaches. We evaluate the learned distance metric by the classification error of a k-NN on the test data, where the number of nearest neighbors k is set to be 3 based on our experience.

Parameter R in the proposed algorithms determines the domain size for the distance metric to be learned. We observe that the classification error of k-NN remains almost unchanged when varying R in the range of $\{100, 1000, 10000\}$. We thus set $R = 1,000$ for all the experiments. Another important parameter used by the proposed algorithms is the step size η. We evaluate the impact of step size η by measuring the classification error of a k-NN algorithm that uses the distance metric learned by the Mini-SGD algorithm with $\eta = \{0.1, 1, 10\}$. We observe that $\eta = 1$ yields a low classification error for almost all datasets by cross-validation with $R = 1,000$ and $T = 10$. We thus fix $\eta = 1$ for the proposed algorithms in all the experiments.

4.2 Experiment (I): Effectiveness of the Proposed SGD Algorithms for DML

In this experiment, we compare the performance of the proposed SGD algorithms for DML, i.e., Mini-SGD, AS-SGD and two hybrid approaches (HR-SGD and HA-SGD), to the full version of SGD for DML (SGD). We also include Euclidean distance as the reference method in our comparison. Table 2 shows the classification error of k-NN ($k = 3$) using the distance metric learned by different DML algorithms. First, it is not surprising to observe that all the distance metric learning algorithms improve the classification performance of k-NN compared to the Euclidean distance. Second, for almost all datasets, we observe that all the proposed DML algorithms (i.e., Mini-SGD, AS-SGD, HR-SGD, and HA-SGD) yield similar classification performance as SGD, the full version of SGD algorithm for DML. This result confirms that the proposed SGD algorithms are effective for DML despite the modifications we made to the SGD algorithm.

4.3 Experiment (II): Efficiency of the Proposed SGD Algorithms for DML

Table. 3 summarizes the running time for the proposed DML algorithms and the SGD method. We note that the running time in Table 3 does not take into account the time for constructing triplet constraints since it is shared by all the methods in comparison.

It is not surprising to observe that all the proposed SGD algorithms, including Mini-SGD, AS-SGD, HA-SGD and HR-SGD, significantly reduce the running time of SGD. For instance, for dataset *isolet*, it takes SGD more than $32,000$ seconds to learn a distance metric, while the running time is reduced to less than $3,500$ seconds when applying the proposed SGD algorithms, roughly a factor of 10 reduction in running time. Comparing the running time of AS-SGD to that of Mini-SGD, we observe that each method has its own advantage: AS-SGD is more efficient on datasets *semeion*, *dna*, *isolet*, and *tdt30*, while Mini-SGD is more efficient on the other six datasets. This is because different mechanisms are employed by AS-SGD and Mini-SGD to reduce the computational cost: AS-SGD improves the computational efficiency of DML by skipping the constraints that are easy to be classified, while Mini-SGD improves the the computational efficiency of SGD by performing the updating of distance metric once for multiple triplet constraints. Finally, we observe that the two hybrid approaches that combine the strength of both adaptive sampling and mini-batch SGD, are computationally most efficient for almost all datasets. We also observe that HR-SGD appears to be more efficient than HA-SGD on six datasets and only loses on datasets *protein*, *sensit* and *rcv20*. This is because HR-SGD computes the sampling probability γ_t based on one randomly sampled triplet while HA-SGD needs to compute the average derivative for each mini-batch of triplet constraints for the sampling probability.

To further examine the computational efficiency of proposed SGD algorithms for DML, we summarize in Table 4 the number of updating performed by different SGD algorithms. We observe that all the proposed SGD algorithms for DML are able to reduce the number of updates significantly compared to SGD. Comparing Mini-SGD to AS-SGD, we observe that for some datasets (e.g., *semeion*, *dna*, *isolet*, and *tdt30*), the number of updates performed by AS-SGD is significantly less than Mini-SGD, while it is the other way around for the other datasets. This is again due to the fact that AS-SGD and Mini-SGD deploy different mechanisms for reducing computational costs. As we expect, the two hybrid approaches are able to further reduce the number of updates performed by AS-SGD and Mini-SGD, making them more efficient algorithms for DML.

By comparing the results in Table 3 to the results in Table 4, we observe that a small number of updates does NOT always guarantee a short running time. This is exhibited by the comparison between the two hybrid approaches: although HA-SGD performs the similar number of updates as HR-SGD on datasets *dna* and *isolet*, it takes HA-SGD significantly longer time to finish the computation than HR-SGD. This is also exhibited by comparing the results across different datasets for a fixed method. For example, for the HA-SGD method, the number of updates for the *protein* dataset is nearly the same as that for the *poker* dataset, but the running time for the *protein* dataset is about 50 times longer than that for the *poker* dataset. This result may sound counter intuitive at the first glance. But, a more careful analysis reveals that in addition to the number of updates, the running time of DML is also affected by the computational cost per iteration, which explains the consistency between Table 3 and 4. In the case of comparing the two hybrid approaches, we observe that HA-SGD is subjected to a higher computational cost per iteration than HR-SGD because HA-SGD has to compute the norm of the *average* gradient over each mini-batch while HR-SGD only needs to compute the derivative of *one* randomly sampled triplet constraint for each mini-batch. In the case of comparing the running time across different datasets, the *protein* dataset has a significantly higher dimensionality than the *poker* dataset, and therefore is subjected to a higher computational cost per iteration because the computational cost of projecting an updated distance metric onto the PSD cone increases at least quadratically in the dimensionality.

Table 2: Classification error (%) of k-NN ($k = 3$) using the distance metrics learned by different SGD methods, online learning algorithms and batch learning approach for DML.

	Baseline	Batch	Online Learning				Proposed Methods			
	Euclidean	LMNN	LEGO	OASIS	SPML	SGD	Mini-SGD	AS-SGD	HR-SGD	HA-SGD
semeion	8.7	9.0	11.9	8.3	6.3	6.3	6.5	6.3	6.4	6.2
dna	20.7	6.2	9.3	16.6	9.1	8.6	9.4	8.4	8.1	8.1
isolet	9.0	5.4	8.3	6.5	6.6	6.3	6.2	6.0	6.4	6.1
tdt30	5.3	3.0	14.6	4.0	3.7	3.8	3.7	3.7	3.8	3.6
letter	4.4	3.2	4.0	2.2	3.1	2.1	2.5	2.1	2.5	2.3
protein	50.0	40.1	42.4	40.1	41.9	40.7	38.9	40.7	41.0	40.9
connect4	29.5	21.1	25.8	22.1	24.5	20.1	20.1	20.1	22.2	20.4
sensit	27.3	24.3	25.4	24.1	23.7	24.0	24.0	24.0	24.4	24.6
rcv20	9.1	N/A	8.9	8.6	8.9	8.5	8.7	8.4	8.4	8.6
poker	38.0	N/A	39.2	36.1	37.8	35.0	33.8	35.0	34.3	34.4

Table 3: Running time (seconds) for different SGD methods, online learning algorithms and batch learning approach for DML. Note that LMNN, a batch DML algorithm, is mainly implemented in C, while the other algorithms in comparison are implemented in Matlab, which is usually less efficient than C.

	Batch	Online Learning				Proposed Methods			
	LMNN	LEGO	OASIS	SPML	SGD	Mini-SGD	AS-SGD	HR-SGD	HA-SGD
semeion	112.7	355.8	29.1	206.6	2,172.4	263.2	45.2	7.4	42.4
dna	255.9	330.2	39.1	122.1	1,165.3	121.0	30.6	7.1	28.0
isolet	2,454.3	3,454.2	515.7	3,017.2	32,762.7	3,440.7	908.4	127.6	246.3
tdt30	264.5	372.6	51.2	145.1	1,351.0	148.0	108.8	11.6	41.6
letter	251.6	15.0	10.8	5.6	27.3	5.3	10.9	1.8	3.2
protein	3,906.4	1,318.9	3,825.9	573.8	5,448.9	580.6	1,335.8	184.5	145.6
connect4	540.2	23.1	79.0	16.4	109.6	15.9	60.5	8.0	6.97
sensit	10,481.2	93.3	303.9	44.3	365.4	41.3	243.9	26.2	17.9
rcv20	N/A	443.6	1,313.7	154.4	1,542.1	158.4	932.9	101.4	45.8
poker	N/A	17.3	17.6	5.8	21.0	4.5	13.5	2.8	3.4

4.4 Experiment (III): Comparison with State-of-the-art Online DML Methods

We compare the proposed SGD algorithms to three state-of-the-art online algorithms and one bath method for DML:

- **SPML** [18]: an online learning algorithm for DML that is based on mini-batch SGD and the hinge loss,
- **OASIS** [7]: a state-of-the-art online DML algorithm,
- **LEGO** [16]: an online version of the information theoretic based DML algorithm [9].

Finally, for sanity checking, we also compare the proposed SGD algorithms to **LMNN** [19], a state-of-the-art batch learning algorithm for DML.

Both SPML and OASIS use the same set of triplet constraints to learn a distance metric as the proposed SGD algorithms. However, unlike SPML and OASIS, pairwise constraints are used by LEGO for DML. For fair comparison, we generate the pairwise constraints for LEGO by splitting each triplet constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t, \mathbf{x}_k^t)$ into two pairwise constraints: a must-link constraint $(\mathbf{x}_i^t, \mathbf{x}_j^t)$ and a cannot-link constraint $(\mathbf{x}_i^t, \mathbf{x}_k^t)$. This splitting operation results in a total of $200,000$ pairwise constraints for LEGO. Finally, we note that since LMNN is a batch learning method, it is allowed to utilize *any* triplet constraint derived from the data, and is not restricted to the set of triplet constraints we generate for the SGD methods. All the baseline DML algorithms are implemented by using the codes from the original authors except for SPML, for which we made appropriate changes to the original code in order to avoid large matrix multiplication and improve the computational efficiency. SPML, OASIS and LEGO are implemented in Matlab, while the core parts of LMNN are implemented by C that is usually deemed to be more efficient than Matlab. The default parameters suggested by the original authors are used in the baseline algorithms. The step size of LEGO is set to be 1, as it was observed in [7] that the prediction performance of LEGO is in general insensitive to the step size. In all experiments, all the baseline methods set the initial solution for distance metric to be an identity matrix.

Table. 2 summarizes the classification results of k-NN ($k = 3$) using the distance metrics learned by the four baseline algorithms. First, we observe that LEGO performs significantly worse than the proposed DML algorithms for five datasets, including *semeion*, *isolet*, *tdt30*, *connect4*, and *poker*. This can be explained by the fact that LEGO uses pairwise constraints for DML while the other methods in comparison use triplet constraints for DML. According to [7, 18, 19], triplet constraints are in general more effective than pairwise constraints. Second, although both SPML and Mini-SGD are based on the mini-batch strategy, SPML performs significantly worse than Mini-SGD on three datasets, i.e. *protein*, *connect4*, and *poker*. The performance difference between SPML and Mini-SGD can be explained by the fact that Mini-SGD uses a smooth loss function while a hinge loss is used by SPML. According to our analysis and the analysis in [8], using a smooth loss function is critical for the success of the mini-batch strategy. Third, OASIS yields

Table 4: The number of updates for different SGD methods and online learning algorithms for DML.

	Online Learning				Proposed Methods			
	LEGO	OASIS	SPML	SGD	Mini-SGD	AS-SGD	HR-SGD	HA-SGD
semeion	71,142.4	432.7	10,000	100,000	10,000	142.2	101.4	162.8
dna	140,027	2,042	10,000	100,000	10,000	707	351	372
isolet	110,175	1,426	10,000	100,000	10,000	1,893	353	378
tdt30	131,997.6	2,284.6	10,000	100,000	10,000	5,563.7	567.6	784.6
letter	130,794	28,063	10,000	100,000	10,000	12,931	1,398	457
protein	166,384	64,804	10,000	100,000	10,000	22,127	3,064	1,623
connect4	153,311.6	69,865	10,000	100,000	10,000	44,510.8	4,161.2	2,134.3
sensit	162,869	78,223	10,000	100,000	10,000	60,028	5,675	1,281
rcv20	137,246	88,476	10,000	100,000	10,000	60,708	6,095	779
poker	179,714	71,620	10,000	100,000	10,000	43,259	4,111	1,635

similar performance as the proposed algorithms for almost all datasets except for datasets *semeion*, *dna* and *poker*, for which OASIS performs significantly worse. Overall, we conclude that the proposed DML algorithms yield similar, if not better, performance as the state-of-the-art online learning algorithms for DML.

Compared to LMNN, a state-of-the-art batch learning algorithm for DML, we observe that the proposed SGD algorithms yield similar performance on three datasets. They however perform significantly better than LMNN on datasets *semeion* and *letter*, and significantly worse on datasets *dna*, *isolet* and *tdt30*. We attribute the difference in classification error to the fact that the proposed DML algorithms are restricted to $100,000$ randomly sampled triplet constraints while LMNN is allowed to use *all* the triplet constraints that can be derived from the data. The restriction in triplet constraints could sometimes limit the classification performance but at the other time help avoid the overfitting problem. We also observe that LMNN is unable to run on the two large datasets *rcv20* and *poker*, indicating that LMNN does not scale well to the size of datasets.

The running time and the number of updates of the baseline online DML algorithms can be found in Table 3 and Table 4, respectively. It is not surprising to observe that the three online DML algorithms are significantly more efficient than SGD in terms of both running time and the number of updates. We also observe that Mini-SGD and SPML share the same number of updates and similar running time for all datasets because they use the same mini-batch strategy. Furthermore, compared to the three online DML algorithms, the two hybrid approaches are significantly more efficient in both running time and the number of updates. Finally, since LMNN is implemented by C, it is not surprising to observe that LMNN shares similar running time as the other online DML algorithms for relatively small datasets. It is however significantly less efficient than the online learning algorithms for datasets of modest size (e.g. *connect4* and *sensit*), and becomes computationally infeasible for the two large datasets *rcv20* and *poker*. Overall, we observe that the two hybrid approaches are significantly more efficient than the other DML algorithms in comparison.

5. CONCLUSION

In this paper, we propose two strategies to improve the computational efficiency of SGD for DML, i.e. mini-batch and adaptive sampling. The key idea of mini-batch is to group multiple triplet constraints into a mini-batch, and only update the distance metric once for each mini-batch; the key idea of adaptive sampling is to perform stochastic updating by giving a difficult triplet constraint more chance to be used for updating the distance metric than an easy triplet constraint. We develop theoretical guarantees for both strategies. We also develop two variants of hybrid approaches that combine mini-batch with adaptive sampling for more efficient DML. Our empirical study confirms that the proposed algorithms yield similar, if not better, prediction performance as the state-of-the-art online learning algorithms for DML but with significantly less amount of running time. Since our empirical study is currently limited to datasets with relatively small number of features, we plan to examine the effectiveness of the proposed algorithms for DML with high dimensional data.

6. REFERENCES

[1] R. Bekkerman and M. Scholz. Data weaving: scaling up the state-of-the-art in data clustering. In *CIKM*, pages 1083–1092, 2008.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[3] D. Cai, X. Wang, and X. He. Probabilistic dyadic data analysis with local and global consistency. In *ICML*, pages 105–112, 2009.

[4] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games.* Cambridge University Press, 2006.

[5] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM TIST*, 2(3):27, 2011.

[6] H. Chang and D.-Y. Yeung. Locally linear metric adaptation for semi-supervised clustering. In *ICML*, pages 153–160, 2004.

[7] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 11:1109–1135, 2010.

[8] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, pages 1647–1655, 2011.

[9] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216, 2007.

[10] M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *J. Parallel Distrib. Comput.*, 64(7):826–838, 2004.

[11] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[12] A. Globerson and S. T. Roweis. Metric learning by collapsing classes. In *NIPS*, page 451, 2005.

[13] E. Hazan and S. Kale. Projection-free online learning. In *ICML*, 2012.

[14] X. He, W.-Y. Ma, and H. Zhang. Learning an image manifold for retrieval. In *ACM Multimedia*, pages 17–23, 2004.

[15] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Trans. on Neural Netw.*, 13(2):415–425, 2002.

[16] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. In *NIPS*, pages 761–768, 2008.

[17] M. Mahdavi, T. Yang, R. Jin, S. Zhu, and J. Yi. Stochastic gradient descent with only one projection. In *NIPS*, pages 503–511, 2012.

[18] B. Shaw, B. C. Huang, and T. Jebara. Learning a distance metric from a network. In *NIPS*, pages 1899–1907, 2011.

[19] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.

[20] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, pages 505–512, 2002.

[21] L. Yang and R. Jin. Distance metric learning: a comprehensive survery. 2006.

[22] J. Zhang, R. Jin, Y. Yang, and A. G. Hauptmann. Modified logistic regression: An approximation to SVM and its applications in large-scale text categorization. In *ICML*, pages 888–895, 2003.

[23] T. Zhang and F. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.

APPENDIX

The analysis for Theorem 1 is in the supplementary document [3] and we give the proof for Theorem 2 here. Define:

$$\begin{aligned} C_N &= \textstyle\sum_{t=1}^N |\ell'(M_t)|, \quad X_t = Z_t - |\ell'(M_t)|, \\ \Lambda_N &= \textstyle\sum_{1\le t\le N}^N X_t, \qquad K = \max_{1\le t\le N} X_t \le 1, \\ \sigma_N^2 &= \textstyle\sum_{t=1}^N E[(Z_t - |\ell'(M_t)|)^2] \le \sum_{t=1}^N |\ell'(M_t)| = C_N \end{aligned}$$

Using Berstein inequality for martingales [4], we have:

$$\begin{aligned} & \Pr(\Lambda_N \ge 2\sqrt{C_N\tau} + \sqrt{2}K\tau/3) \\ = \ & \Pr(\Lambda_N \ge 2\sqrt{C_N\tau} + \sqrt{2}K\tau/3, \sigma_N^2 \le C_N, C_N \le N) \\ \le \ & \Pr\begin{pmatrix} \Lambda_N \ge 2\sqrt{C_N\tau} + \sqrt{2}K\tau/3, \sigma_N^2 \le C_N, \\ C_N \le 1/N \end{pmatrix} \\ & + \sum_{i=1}^m \Pr\begin{pmatrix} \Lambda_N \ge 2\sqrt{C_N\tau} + \sqrt{2}K\tau/3, \sigma_N^2 \le C_N, \\ 2^{i-1}/N < C_N \le 2^i/N \end{pmatrix} \\ \le \ & \Pr(C_N \le 1/N) \\ & + \sum_{i=1}^m \Pr\left(\Lambda_N \ge \sqrt{2\frac{2^i}{N}\tau} + \sqrt{2}K\tau/3, \sigma_N^2 \le \frac{2^i}{N}\right) \\ \le \ & \Pr(C_N \le 1/N) + me^{-\tau} \end{aligned}$$

[3] https://sites.google.com/site/zljzju/Supplymentary.pdf

where $m = \lceil \log_2(N^2) \rceil$. By setting $me^{-\tau} = \delta$, with a probability $1 - \delta$, the number of updates can be bounded as:

$$\begin{aligned} \sum_{t=1}^N Z_t &\le C_N + \frac{1}{2}C_N + 2\ln\frac{m}{\delta} + \frac{\sqrt{2}}{3}K\ln\frac{m}{\delta} \\ &\le \frac{3}{2}L\sum_{t=1}^N \ell(M_t) + \frac{5}{2}\ln\frac{m}{\delta} \qquad (7) \end{aligned}$$

Then, we give the regret bound. Using the standard analysis for online learning [4], we have:

$$\begin{aligned} & \ell(M_t) - \ell(M_*) \le \langle \ell'(M_t)A_t, M_t - M_* \rangle \\ = \ & \tau_t Z_t \langle A_t, M_t - M_* \rangle \\ & + (\ell'(M_t) - \tau_t Z_t)\langle A_t, M_t - M_* \rangle \\ \le \ & \frac{\|M_t - M_*\|_F^2 - \|M_{t+1} - M_*\|_F^2}{2\eta} + \frac{\eta A^2 Z_t}{2} \\ & + \tau_t(|\ell'(M_t)| - Z_t)\langle A_t, M_t - M_* \rangle \end{aligned}$$

Taking the sum from $t = 1$ to N, we have:

$$\begin{aligned} & \sum_{t=1}^N \ell(M_t) - \ell(M_*) \le \frac{\|M_1 - M_*\|_F^2}{2\eta} + \frac{\eta A^2}{2}\sum_{t=1}^N Z_t \\ & + \sum_{t=1}^N 2\tau_t(|\ell'(M_t)| - Z_t)RA \end{aligned}$$

According to (7), with a probability $1 - \delta$, the second item could be bounded as:

$$\begin{aligned} \frac{\eta A^2}{2}\sum_{t=1}^N Z_t &\le \eta A^2\left(\frac{3}{4}L\sum_{t=1}^N \ell(M_t) + \frac{5}{4}\ln\frac{m}{\delta}\right) \\ &\le \frac{3}{4}\gamma\sum_{t=1}^N \ell(M_t) + \frac{5}{4}\eta A^2\ln\frac{m}{\delta} \qquad (8) \end{aligned}$$

where $\gamma \ge \eta L A^2$.

Applying Berstein inequality for martingales [4] for the last item, we have, with a probability $1 - \delta$:

$$\begin{aligned} & \textstyle\sum_{t=1}^N 2\tau_t(|\ell'(M_t)| - Z_t)RA \le 4RA\sqrt{C_N \ln\frac{m}{\delta}} + \frac{2\sqrt{2}}{3}RA\ln\frac{m}{\delta} \\ & \le \frac{\gamma}{4}\sum_{t=1}^N \ell(M_t) + \frac{16R^2}{\eta}\ln\frac{m}{\delta} + RA\ln\frac{m}{\delta} \qquad (9) \end{aligned}$$

Combining the bounds in (8) and (9), we have, with a probability $1 - 2\delta$:

$$\begin{aligned} & \sum_{t=1}^N \ell(M_t) - \ell(M_*) \le \frac{1}{2\eta}\left(R^2 + 32R^2\ln\frac{m}{\delta}\right) \\ & + \gamma\sum_{t=1}^N \ell(M_t) + \frac{5}{4}\eta A^2\ln\frac{m}{\delta} + RA\ln\frac{m}{\delta} \end{aligned}$$

which is equal to:

$$\mathcal{L}(\bar{M}) \le \frac{1}{1-\gamma}\left(\mathcal{L}(M_*) + \frac{R^2 c}{\eta N} + \frac{\eta c}{N} + \frac{c}{N}\right)$$

where

$$c = \max\left\{\frac{1}{2} + 16\ln\frac{m}{\delta}, \frac{5}{4}A^2\ln\frac{m}{\delta}, RA\ln\frac{m}{\delta}\right\}$$

The proof is completed by setting $\gamma = 3\eta L A^2$.